HP-CAST, November 15, 2014

Dr. Ronald Young

What Will Limit Performance When Floating Point Operations are Free?

HP SL270 Server with (8) NVIDIA K40m GPUs – [Matrix]Warrior benchmark

Performance (Gflops)

Routine	All	CPU's	GPU 0	GPU 1	GPU 2	GPU 3	GPU 4	GPU 5	GPU 6	GPU 7
Matrix Multiply CPU(0%)	9430	0	1197	1201	1188	1210	1203	1206	1181	1185

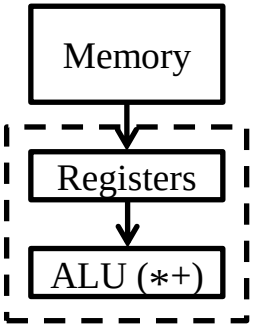
Machine	Year	Flops	$/Gflop
DEC VAX	**1978**	97,000	$2,000,000,000
(8) NVIDIA GPUs	**2014**	9,430,000,000,000	$10
Ratio		~100,000,000	~200,000,000

Computer Evolution

Early Machines

Limited by Floating Point & Storage

Processing speed = Memory speed



Computer Evolution

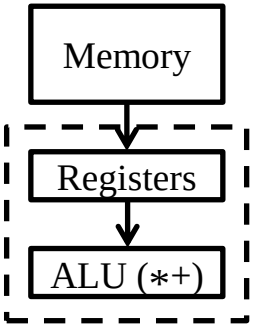
Early Machines

Limited by Floating Point & Storage

Processing speed = Memory speed

↓

RISC Processor



Computer Evolution

Early Machines

Limited by Floating Point & Storage

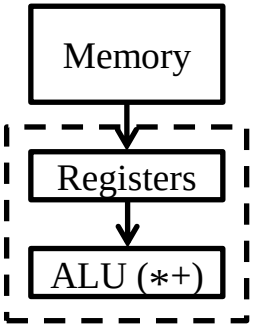
Processing speed = Memory speed

↓

RISC Processor

↓

Processing speed >> Memory speed



Computer Evolution

Early Machines

Limited by Floating Point & Storage

Processing speed = Memory speed

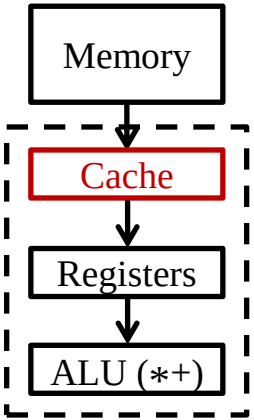
↓

RISC Processor

↓

Hierarchical Memory Systems

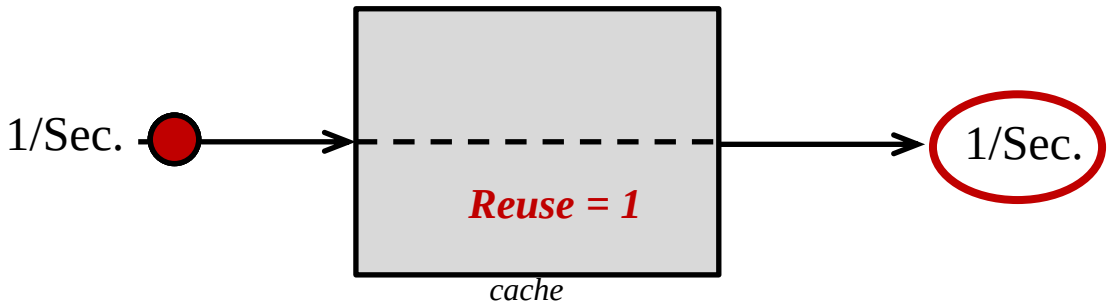
Processing speed >> Memory speed



Reuse *Efficiently linking slower input to faster requirement*

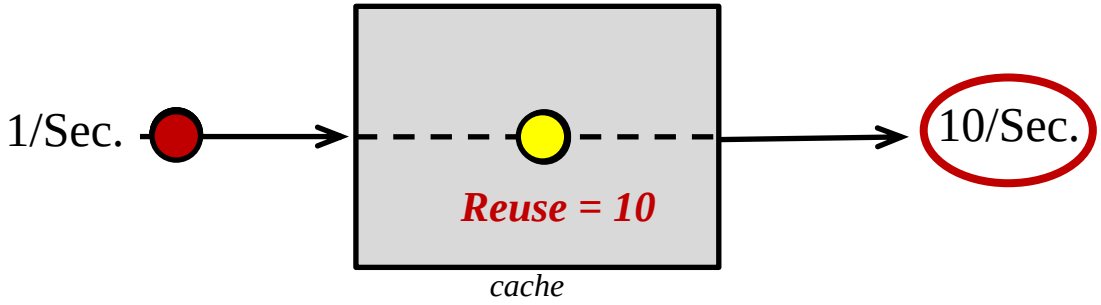
- ***Reuse = Number of times a data element is used in a computation.***

Old machines: Low Reuse



New machines: **High Reuse**

Just an example: can be much higher than 10



Computer Evolution

Early Machines

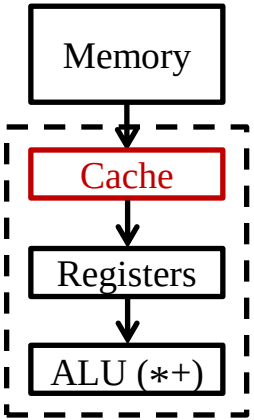
Limited by Floating Point & Storage

Processing speed = Memory speed

RISC Processor

Hierarchical Memory Systems

Processing speed >> Memory speed



Reuse

Low Reuse

Limited by Memory Bandwidth

High Reuse

Limited by Floating Point

Computer Evolution

Early Machines

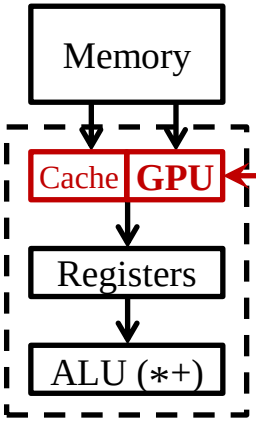
Limited by Floating Point & Storage

Processing speed = Memory speed

RISC Processor

Hierarchical Memory Systems

Processing speed >> Memory speed



Reuse

Low Reuse

Limited by Memory Bandwidth

High Reuse

Limited by Floating Point

Computer Evolution

Early Machines

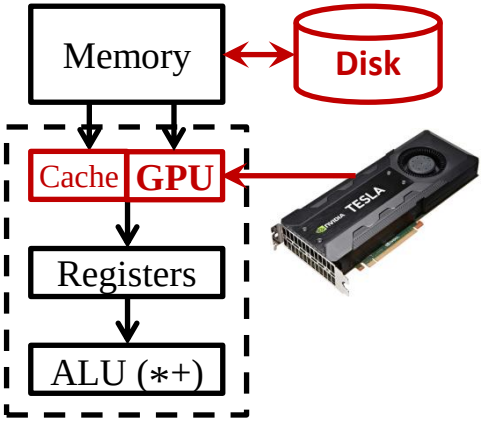
Limited by Floating Point & Storage

Processing speed = Memory speed

RISC Processor

Hierarchical Memory Systems

Processing speed >> Memory speed



Reuse

Low Reuse

Limited by Memory Bandwidth

High Reuse

Limited by Floating Point

Computer Evolution

Early Machines

Limited by Floating Point & Storage

Processing speed = Memory speed

RISC Processor

Hierarchical Memory Systems

Processing speed >> Memory speed

Reuse

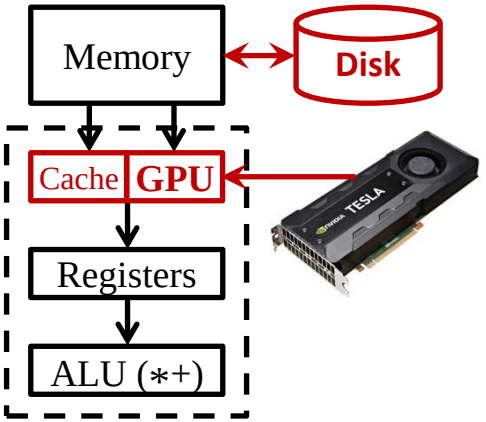
Low Reuse

Limited by Memory Bandwidth

Migrate

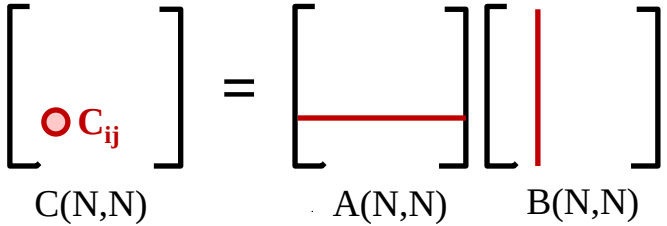
High Reuse

Limited by Floating Point



BLAS 1 Reuse

When M=N=K

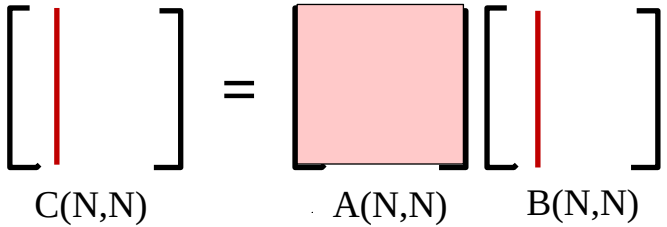


BLAS	Function	Data	Flops	Reuse
1	DDOT	2N+1	2N	**1**

```
      DO 30 J=1,N
       DO 20 I=1,M
        DO 10 L=1,K
         C(I,J) = A(I,L)*B(L,J)
10      CONTINUE
20     CONTINUE
30    CONTINUE
```

$$\frac{\text{Flops}}{\text{Data}} = \frac{2N}{(2N+1)} \sim 1$$

BLAS 2 Reuse

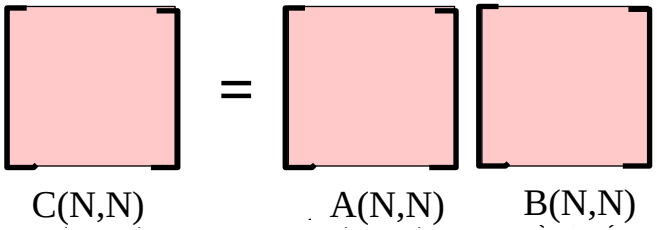


BLAS	Function	Data	Flops	Reuse
1	DDOT	2N+1	2N	**1**
2	DGEMV	N^2+2N	$2N^2$	**2**

```
      DO 30 J=1,N
       DO 20 I=1,M
        DO 10 L=1,K
         C(I,J) = A(I,L)*B(L,J)
10      CONTINUE
20     CONTINUE
30    CONTINUE
```

$$\frac{\text{Flops}}{\text{Data}} = \frac{2N^2}{(N^2+2N)} \sim 2$$

BLAS 3 Reuse

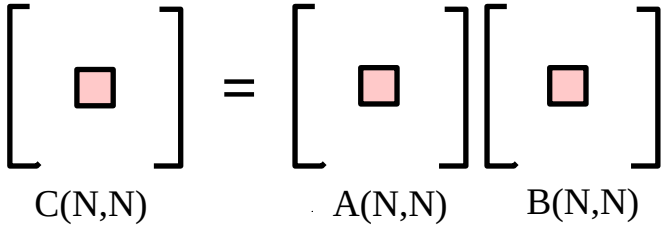


BLAS	Function	Data	Flops	Reuse
1	DDOT	2N+1	2N	**1**
2	DGEMV	N^2+2N	$2N^2$	**2**
3	DGEMM	$3N^2$	$2N^3$	**2N/3**

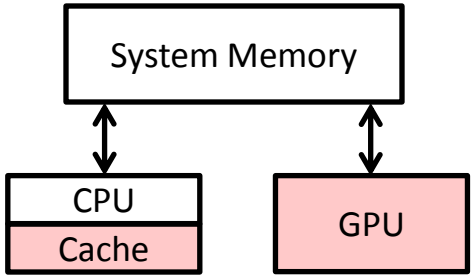
```
   DO 30 J=1,N
    DO 20 I=1,M
     DO 10 L=1,K
      C(I,J) = A(I,L)*B(L,J)
10    CONTINUE
20  CONTINUE
30 CONTINUE
```

$$\frac{\text{Flops}}{\text{Data}} = \frac{2N^3}{3N^2} = \frac{2N}{3}$$

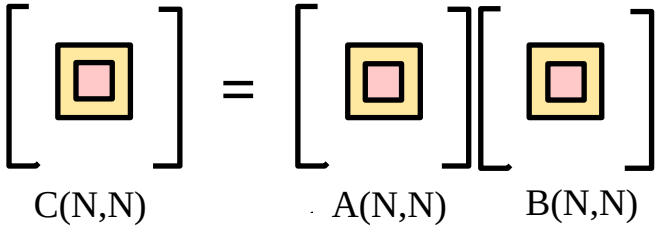
Reuse – Small Data



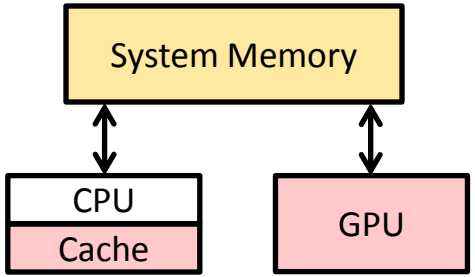
BLAS	Function	Data	Flops	Reuse
1	DDOT	2N+1	2N	**1**
2	DGEMV	N^2+2N	$2N^2$	**2**
3	DGEMM	$3N^2$	$2N^3$	**2N/3**



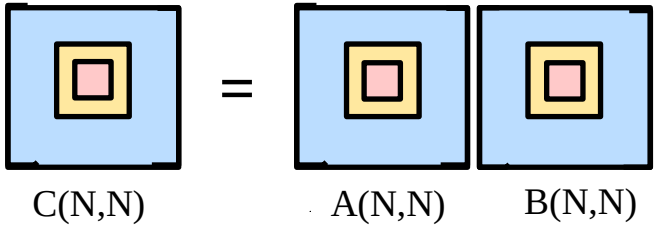
Reuse – More Data



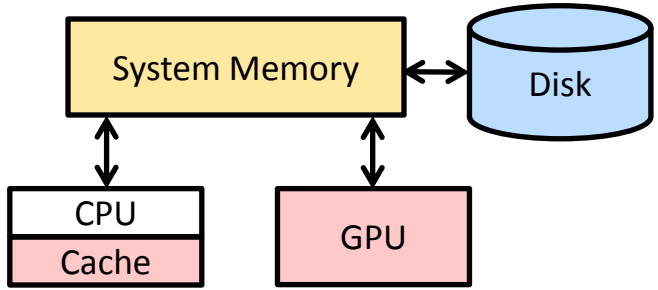
BLAS	Function	Data	Flops	Reuse
~~1~~	~~DDOT~~	~~2N+1~~	~~2N~~	~~**1**~~
~~2~~	~~DGEMV~~	~~N^2+2N~~	~~$2N^2$~~	~~**2**~~
3	DGEMM	$3N^2$	$2N^3$	**2N/3**



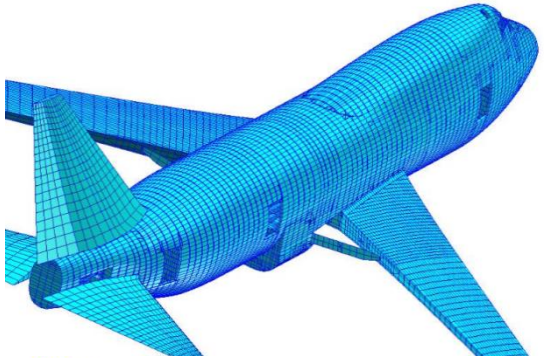
Reuse – Big Data

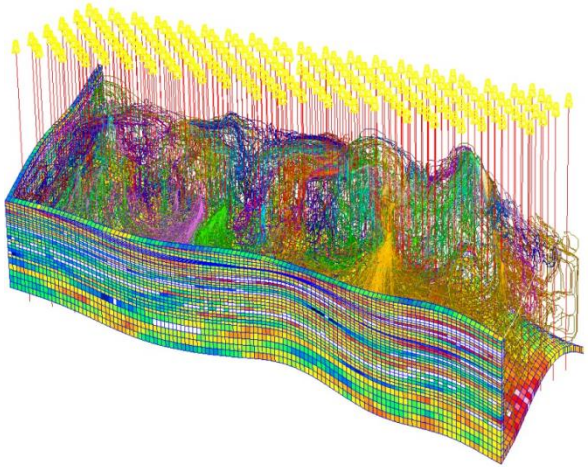


BLAS	Function	Data	Flops	Reuse
~~1~~	~~DDOT~~	~~2N+1~~	~~2N~~	~~**1**~~
~~2~~	~~DGEMV~~	~~N^2+2N~~	~~$2N^2$~~	~~**2**~~
3	DGEMM	$3N^2$	$2N^3$	**2N/3**



Application Reuse





Discretization

1. Domain is divided into a large number of pieces.
2. Equations are formed for each piece **[a], {b}**
3. Pieces are assembled into a global system of equations

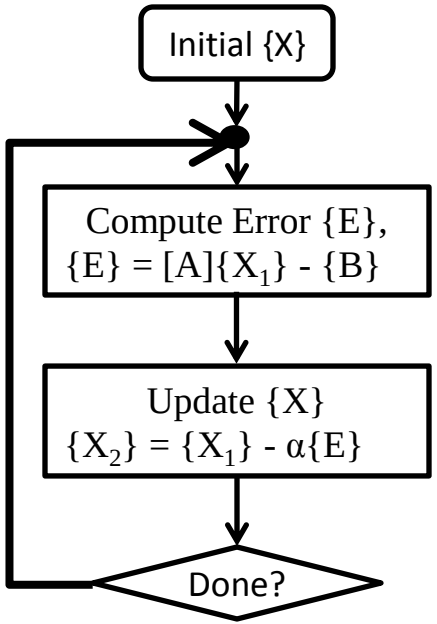
   [A]{X} = {B}
4. The system is then solved for the solution **{X}**

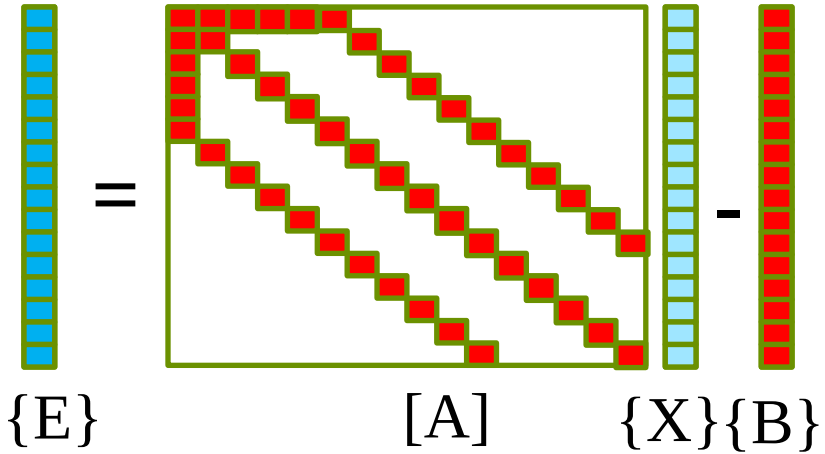
More pieces gives better answer

More pieces = more data and processing

∴ Machine (speed, storage) determines accuracy

Low reuse Solving [A]{X}={B} ***Explicitly***





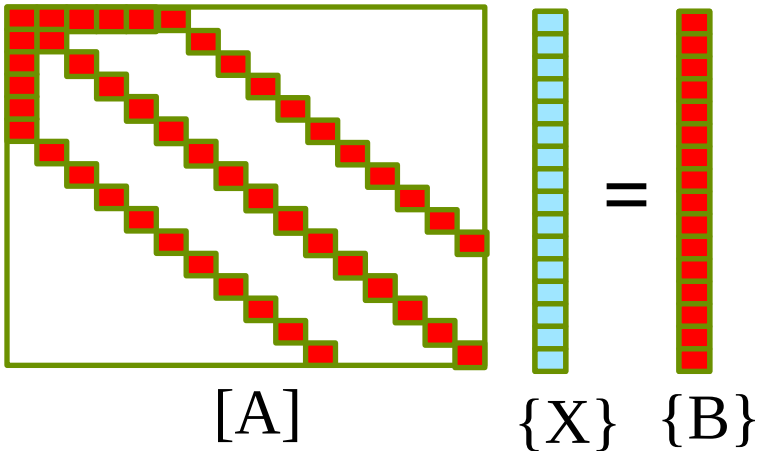
- **(+)** Minimum data
- **(-)** Low reuse: BLAS 2
- **(-)** Indirect addressing
- **(-)** Limited by memory size
- **(-)** Many iterations
- **(-)** Convergence risk
- **(-)** Data dependent
- **(-)** Unpredictable run times
- **(-)** Multiple solutions
- **(-)** Stability -> small increments

High reuse

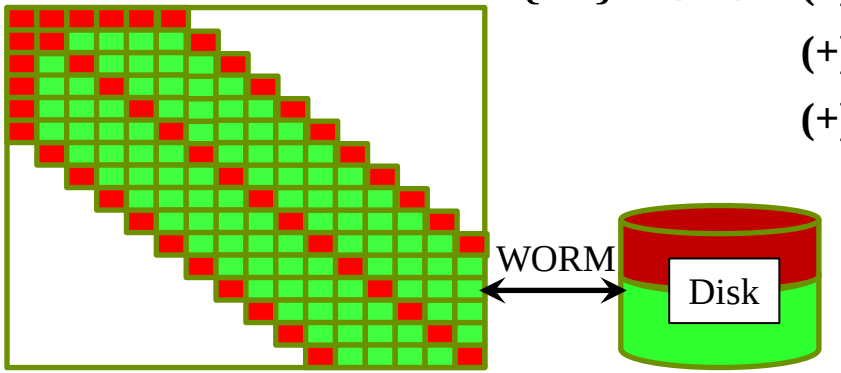
Solving [A]{X}={B} *Implicitly*

Factor [A]
Symmetric
$[L][D][L]^T$
Nonsymmetric
[L][U]

Solve
Symmetric
$[L][D][L]^T$ {X}= {B}
Nonsymmetric
[L][U] {X} = {B}

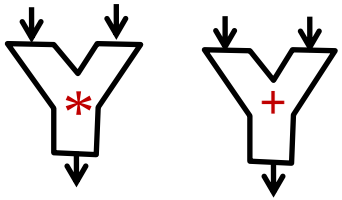


- **(-)** Fill data
- **(+)** High reuse: BLAS 3
- **(+)** Incremental addressing
- **(+)** One iteration
- **(+)** No convergence risk
- **(+)** Data value independent
- **(+)** Predictable run times
- **(+)** Multiple solutions
- **(+)** Unconditionally stable

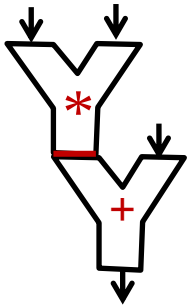


Solid State Disk (SSD)
and
Shingled Magnetic Rec.(SMR)

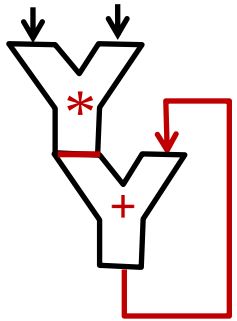
Hardware Reuse



(6)(8)/2 = 24 Bytes/flop

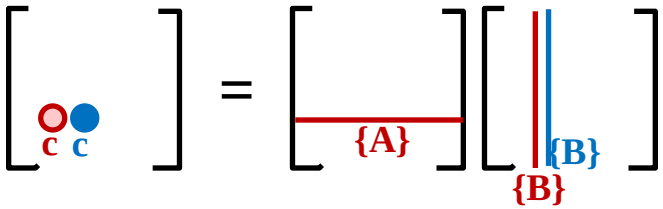


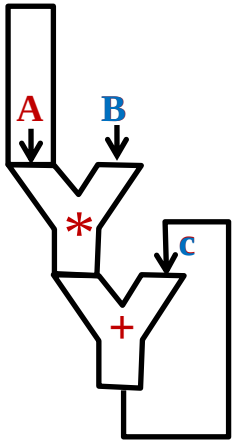
Fused Multiply-Add
2008: IEEE 754
(4)(8)/2 = 16 Bytes/flop



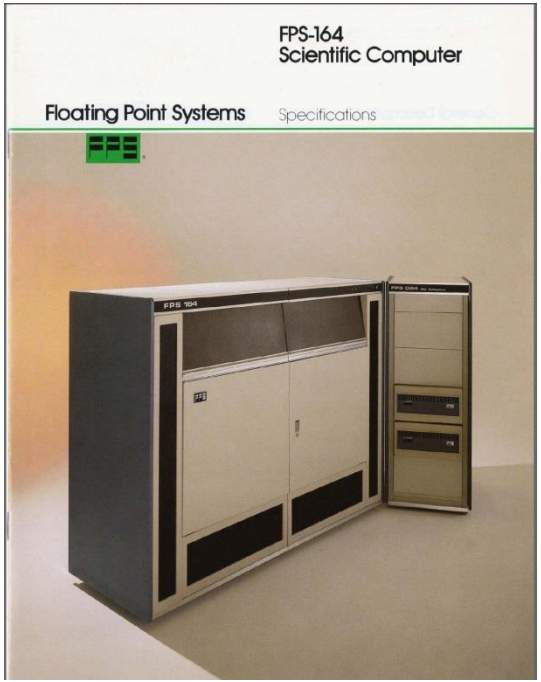
Dot Product
(2)(8)/2 = 8 Bytes/flop

Hardware Reuse



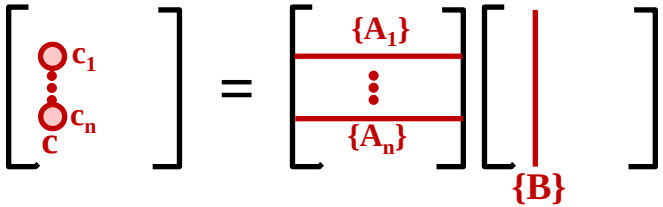


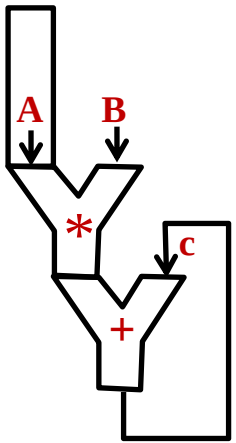
Vector Dot Product
(1)(8)/2 = 4 Bytes/flop



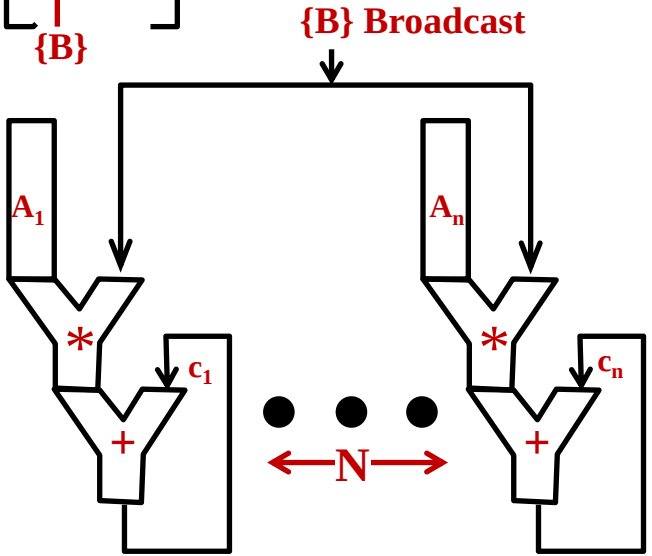
Floating Point Systems 164
First 64-bit accelerator
First version of FMSlib

Hardware Reuse

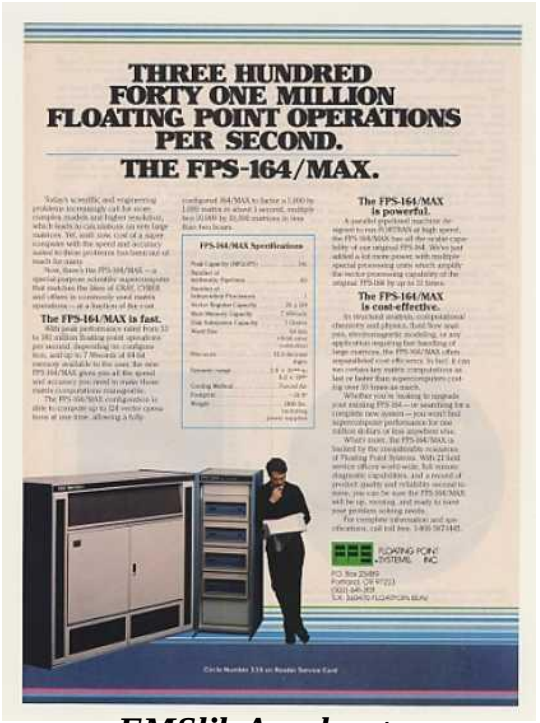




Vector Dot Product
(1)(8)/2 = 4 Bytes/flop

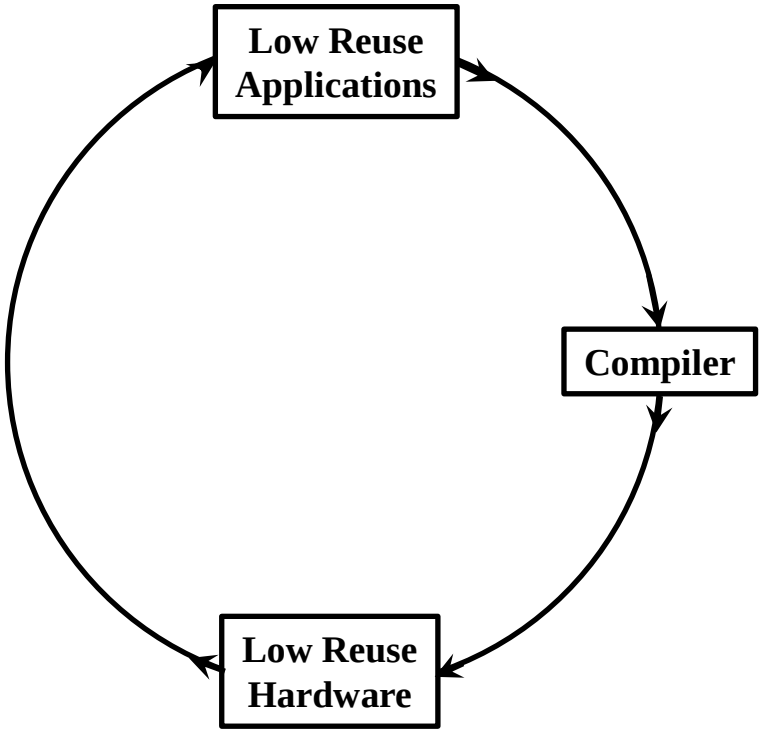


Parallel Vector Dot Product
(1/N)(4) = (4/N) Bytes/flop

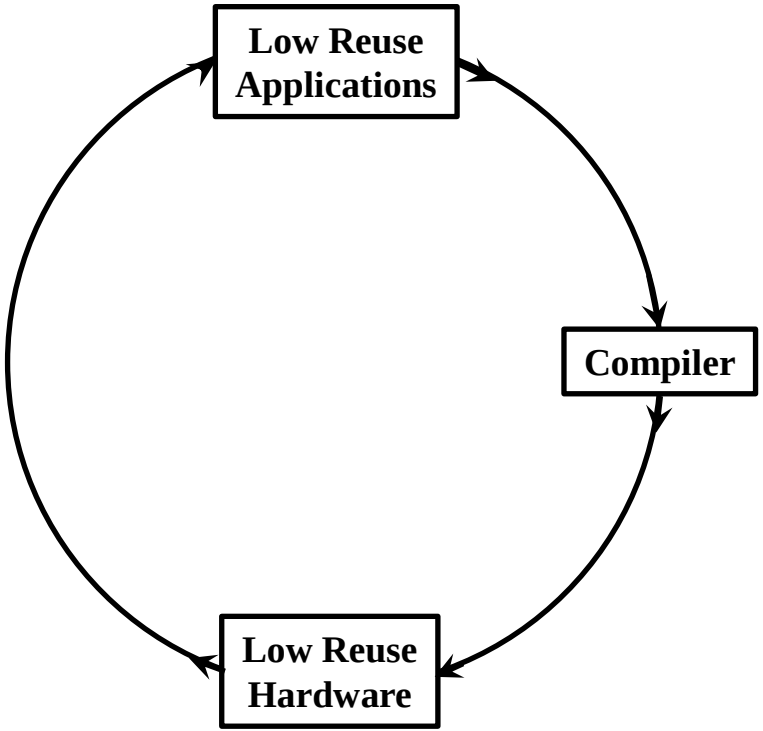

THREE HUNDRED FORTY ONE MILLION FLOATING POINT OPERATIONS PER SECOND.
THE FPS-164/MAX.

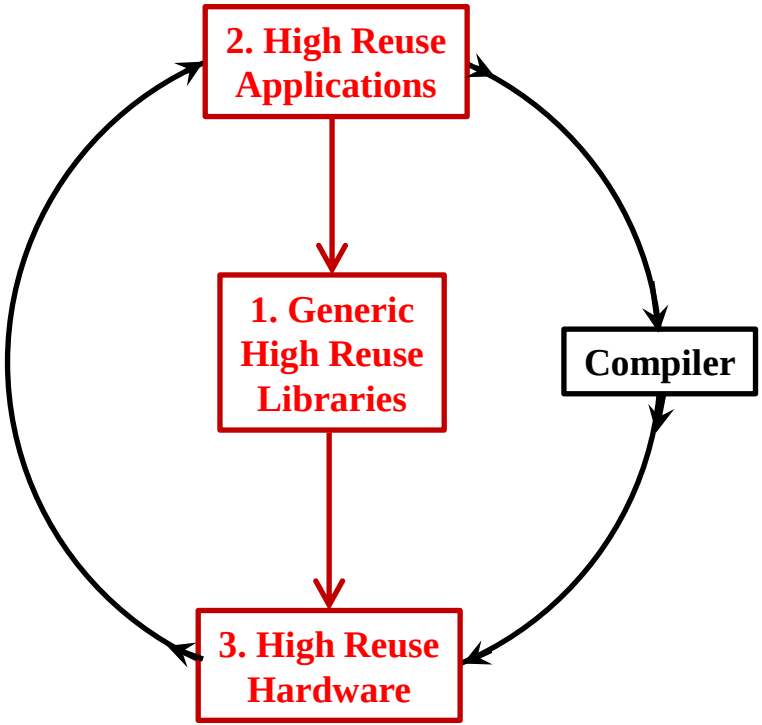
FMSlib Accelerator

Migrating to **High Reuse**

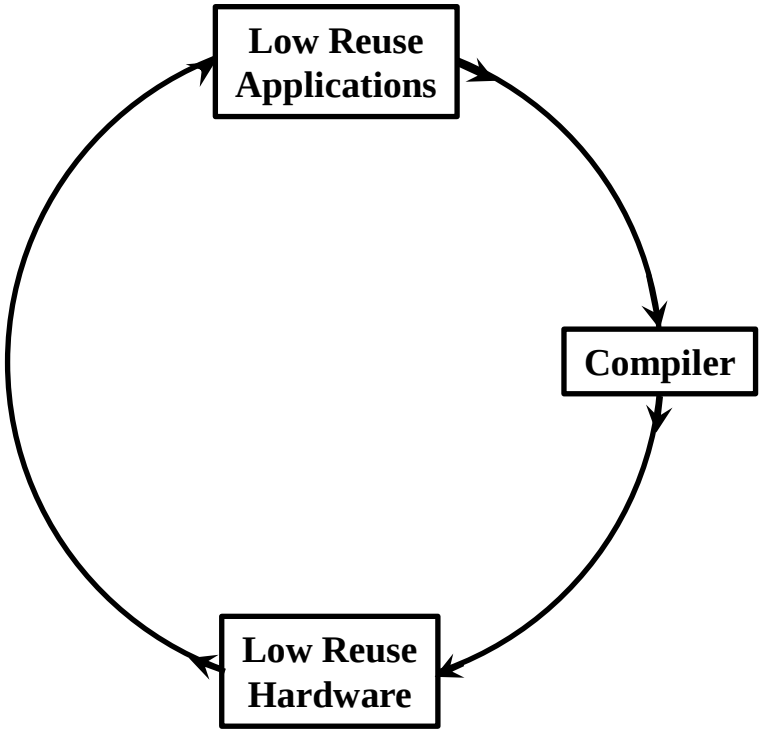


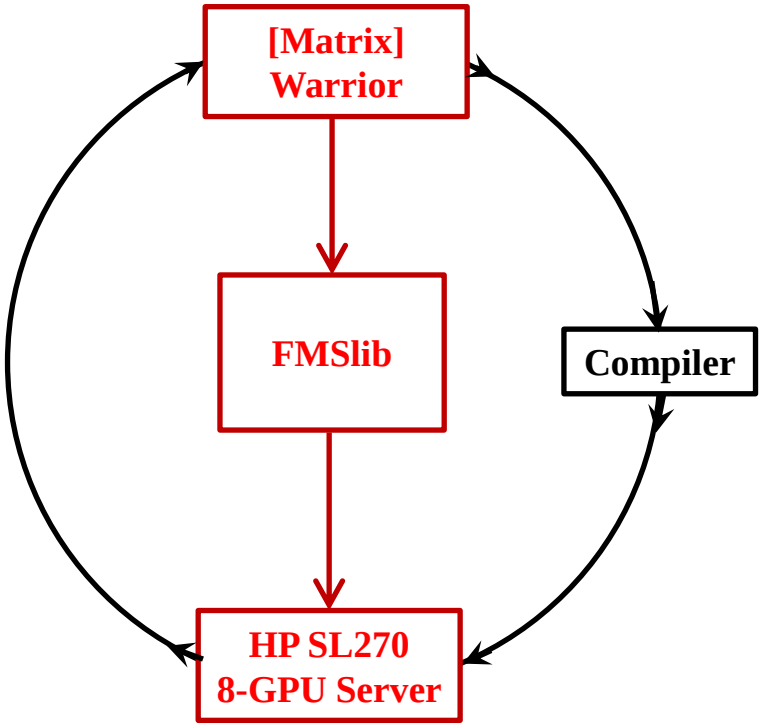
Migrating to **High Reuse**





Migrating to **High Reuse**



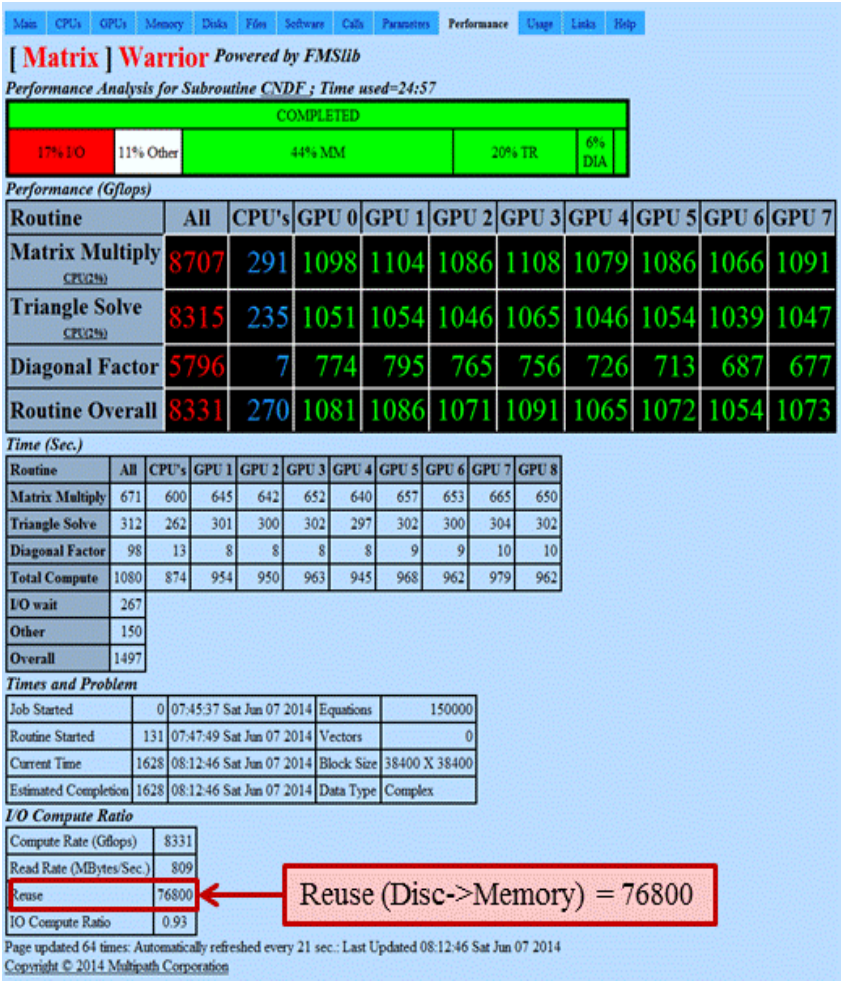


Explore Reuse

Download and run [Matrix]Warrior, an application which benchmarks and demonstrates FMSlib

[Matrix]Warrior
FMSlib

- *Form **[A]**, **{B}** with test data*
- *Use FMSlib to solve* ***[A]{X}={B}***
- *Display reports as WEB pages*



Main | CPUs | GPUs | Memory | Disks | Files | Software | Calls | Parameters | Performance | Usage | Links | Help

[Matrix] Warrior *Powered by FMSlib*

Performance Analysis for Subroutine CNDF ; Time used=24:57

COMPLETED

17% I/O | 11% Other | 44% MM | 20% TR | 6% DIA

Performance (Gflops)

Routine	All	CPU's	GPU 0	GPU 1	GPU 2	GPU 3	GPU 4	GPU 5	GPU 6	GPU 7
Matrix Multiply CPU(2%)	8707	291	1098	1104	1086	1108	1079	1086	1066	1091
Triangle Solve CPU(2%)	8315	235	1051	1054	1046	1065	1046	1054	1039	1047
Diagonal Factor	5796	7	774	795	765	756	726	713	687	677
Routine Overall	8331	270	1081	1086	1071	1091	1065	1072	1054	1073

Time (Sec.)

Routine	All	CPU's	GPU 1	GPU 2	GPU 3	GPU 4	GPU 5	GPU 6	GPU 7	GPU 8
Matrix Multiply	671	600	645	642	652	640	657	653	665	650
Triangle Solve	312	262	301	300	302	297	302	300	304	302
Diagonal Factor	98	13	8	8	8	8	9	9	10	10
Total Compute	1080	874	954	950	963	945	968	962	979	962
I/O wait	267									
Other	150									
Overall	1497									

Times and Problem

Job Started	0	07:45:37 Sat Jun 07 2014	Equations	150000
Routine Started	131	07:47:49 Sat Jun 07 2014	Vectors	0
Current Time	1628	08:12:46 Sat Jun 07 2014	Block Size	38400 X 38400
Estimated Completion	1628	08:12:46 Sat Jun 07 2014	Data Type	Complex

I/O Compute Ratio

Compute Rate (Gflops)	8331
Read Rate (MBytes/Sec.)	809
Reuse	76800
IO Compute Ratio	0.93

Reuse (Disc->Memory) = 76800

Page updated 64 times: Automatically refreshed every 21 sec.: Last Updated 08:12:46 Sat Jun 07 2014

Q: What Will Limit Performance
When Floating Point Operations are Free?

1. Compiled low reuse software
2. Lack of generic high reuse libraries
3. Lack of high reuse hardware

More Information:

fmslib.com

FREE Download
[Matrix]Warrior

This Presentation

NVIDIA GTC Presentations